EECS 3311 Software Design

Winter 2025

Sections M and Z

Deliverable 1 (Total Marks 100)

Overview

In this project, you will learn to design software from scratch, i.e., analyze software requirements and use cases, produce sequence diagrams as a requirements specification, decompose a complex problem into sub-problems, start understanding and addressing software architecture design.

Ask questions on course slack channel or emails.

Group

Each student will be assigned to a group of five students. Please see eClass to get information of the class. You can ask the TAs if you would like to change a group. However, it may not be always possible to change your group if the other group does not want to change.

Policies

Your (submitted or un-submitted) solution to this assignment (which is not revealed to the public) remains the property of the EECS department. Do not distribute or share your code in any public media (e.g., a non-private Github repository) in any way, shape, or form. The department reserves the right to take necessary actions upon found violations of this policy.

- You are highly encouraged to work in a group for this project deliverable.
- When you submit your solution, you claim that it is solely your work. Therefore, it is considered as an violation of academic integrity if you copy or share any parts of your code or documentation.
- When assessing your submission, the instructor and TA may examine your doc/code, and suspicious submissions will be reported to the department/faculty if necessary. We do not tolerate academic dishonesty, so please obey this policy strictly.
- Emailing your solutions to the instruction or TAs will not be acceptable.

Submitting Your Work on eClass

Zip your submission named with the assigned team id of your group. A 2% penalty will be applied if not follow this rule.

The “system-to-be”: YorkU Parking Booking System

“YorkU Parking Management Team is now seeking a new booking system that can help them provide better online booking services to their clients (e.g., students, faculty members, non-faculty staffs, and

visitors). The system is supposed to be a GUI-based Java project. The basic requirements of the system (from an interview with their management teams) are as follows:

- Req1: Any client should be able to register as a user of the system with a unique/valid email and strong password (i.e., a combination of uppercase letters, lowercase letters, numbers, and symbols). The system currently allows four types of clients to be registered, i.e., students, faculty members, non-faculty staffs, and visitors, while it's open for new types in the future. If a client registers as a student, a faculty member or a non-faculty staff, her/his registration requires a further validation from the management teams.
- Req2: The system has an auto account generation subsystem, which can generate management accounts (unique names and strong passwords), with which the management teams can login to the system and maintain the parking services. Note that, the auto account generation is only available for the super manager (ONLY ONE) of the Parking team.
- Req3: Any registered client can book a valid parking space (i.e., currently not occupied or booked by other users) after login to the system, the parking rates for different types of users are different, i.e., 5$, 8$, 10$, and 15$ for students, faculty members, non-faculty staffs, and visitors per hour respectively.
- Req4: Booking a parking space requires the cost of an hour (of the type of a client) as the deposit, which will not be refunded if a no-show happens in the first 1 hour of the booked parking period. Otherwise, the deposit will be deducted when checking out.
- Req5: Suppose each parking space has a sensor to detect if a car is using the parking space or not. In addition, the sensor can also scan the basic info of cars, and further send the essential information to the system.
- Req6: Managers of the system can add, enable, or disable a parking lot, a parking lot contains 100 parking spaces. Managers can also enable or disable a parking space due to maintenance issues.
- Req7: Each parking space will have a unique identification number and other details including its location and its parting lot, which will help with the navigation for clients.
- Req8: To book a parking space, clients need to provide a valid licence plate number. A client can edit or cancel her/his bookings before the starting time of a booking.
- Req9: Clients can extend a parking time before the expiration.
- Req10: Clients can pay their parking fee via different payment options, i.e., debit cards, credit cards, mobile payments, etc.

In Deliverable-I, you are asked to analyze the raw user requirements from clients and create use cases, activity diagram, and sequence diagrams to illustrate the requirements with more detailed, after that you are further asked to create a class diagram that meets the requirements. Note, no design patterns are required at Deliverable-I, we will explore the effect of design patterns on your project in Deliverable-II.

Part I: Requirements Eliciting and Modeling (75 points)

Information from the interview might not cover all the requirements, you can make your own assumptions if necessary. Please draw the diagrams carefully and present as more details as possible. A real-world developer should be able to develop the system based on these diagrams.

Task1: Use Case Diagram

Your first task is to identify the main actors in the system (hints: at least three). For each of the actors, please identify its use cases under different scenarios and draw an use case diagram for each scenario. Please also indicate the sources of your use cases (i.e., from which Reqs).

Task2: Activity Diagram

For each of your use case diagrams drew in Task1, please draw the activity diagram accordingly.

Task3: Sequence Diagram

For each of your use case diagrams drew in Task1, please draw at least one sequence diagram accordingly.

Part II: Design (25 points)

The diagrams you drew in Part I can help developers understand the basic requirements and functionalities of the 'system-to-be'. In this part, we will further explore how to design the 'system-to-be'.

Task 1: Design (Class Diagram)

Before we start the implementation task, you are asked to design the system via drawing detailed class diagrams. Please identify all the possible classes (including their attributes, methods, and relations among classes) and further draw the class diagrams of the 'system-to-be'. Note that, using design patterns at current stage is not mandatory, we will explore how we can use design patterns to improve your designs in Project Deliverable II.